TensorFlow Native Compilation

TensorFlow™ Native Compilation Guide

Introduction

This guide will explain how to set up your machine to run the SYCL™ version of TensorFlow™ using ComputeCpp. This guide describes how to build and run TensorFlow 1.9 natively on any device supporting SPIR or SPIR-V. For cross-compiling please read our other guide.

Configuration management

These instructions relate to the following configuration:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Host platform</strong></td>
<td>Ubuntu 16.04, x86_64 architecture (more recent versions of Ubuntu, or other Linux distributions, may work but are not supported)</td>
</tr>
<tr>
<td><strong>GPU</strong></td>
<td>AMD R9 Nano Fury, Intel Gen9 HD Graphics</td>
</tr>
<tr>
<td><strong>TensorFlow</strong></td>
<td>master</td>
</tr>
<tr>
<td><strong>ComputeCpp</strong></td>
<td>Latest</td>
</tr>
<tr>
<td><strong>Python</strong></td>
<td>3.5 (more recent versions of Python may work but are not supported)</td>
</tr>
</tbody>
</table>

Notes:

- For older or newer versions of TensorFlow, please contact Codeplay for build documentation.
- If you are interested in the latest features you may try our experimental branch.
- OpenCL devices other than those listed above may work, but Codeplay does not support them at this time.
- It is strongly recommended to make sure you have a working OpenCL installation before building TensorFlow, see here.
- It is strongly recommended to test your ComputeCpp installation using the Eigen tests before building TensorFlow.
Install an OpenCL driver

Install an OpenCL driver supporting SPIR or SPIR-V depending on your hardware.

Driver amdgpu-pro version 17.50-511655

```
wget --referer http://support.amd.com/
https://www2.ati.com/drivers/linux/ubuntu/amdgpu-pro-17.50-511655.tar.xz
tar xf amdgpu-pro-17.50-511655.tar.xz
./amdgpu-pro-17.50-511655/amdgpu-pro-install --opencl=legacy --headless
```

These options install the driver for compute only, look at the driver documentation if you want to use this driver for graphics too. This version is required as more recent versions have not been verified to work, and may break OpenCL support for features such as SPIR.

Driver Intel NEO version 18.38.11535

This is the latest version tested. Newer versions of the driver may work but could have performance or correctness issues.

```
wget
https://github.com/intel/compute-runtime/releases/download/18.38.11535/intel-opencl_18.38.11535_amd64.deb
sudo dpkg -i intel-opencl_18.38.11535_amd64.deb
```

Arm Mali driver

Please contact Arm to obtain an Arm Mali driver with support for OpenCL 1.2 with SPIR-V. Note that, as of the date of publication, the publicly released Arm Mali drivers available at https://developer.arm.com/products/software/mali-drivers/user-space do not correctly support SPIR-V.

Imagination PowerVR driver

Please contact Imagination to obtain a PowerVR driver with support for OpenCL 1.2 with SPIR.

Verify your OpenCL installation

```
sudo apt update
sudo apt install clinfo
clinfo
```

The output should list at least one platform and one device. The "Extensions" field of the device properties should include cl_khr_spir and/or cl_khr_il_program. If any errors are present, check the installation of the OpenCL driver. * It is important to have this step working correctly, or it is likely that you run into errors later when running TensorFlow. * For example, if the OpenCL driver cannot be found, ensure that LD_LIBRARY_PATH has been set correctly to include the path to
Install dependency packages

```bash
sudo apt update
sudo apt install git cmake gcc build-essential libpython3-all-dev ocl-icd-opencl-dev opencl-headers openjdk-8-jdk python3 python3-dev python3-pip zlib1g-dev
pip install -U --user numpy==1.14.5 wheel==0.31.1 six==1.11.0 mock==2.0.0 enum34==1.1.6
```

Specific python package versions are added here for reference. More recent versions of numpy is known to break some tests in this version of TensorFlow.

The rest of the guide will assume Python 3.5 is used by default. This can be done for example with:

```bash
alias python=python3.5
alias pip=pip3.5
```

Install toolchains

- Register for an account on Codeplay's developer website:
  https://developer.codeplay.com/computecppce/latest/download
- From that page, download the following version: Latest > linux-gnu > x86_64 computecpp-ce-*-x86_64-linux-gnu.tar.gz

Set up an environment variable with the ComputeCpp version so that you can copy and paste the commands below.

For example:

```bash
export CCPP_VERSION=2.2.1
```

```bash
tar -xf computecpp-ce-$(CCPP_VERSION)-x86_64-linux-gnu.tar.gz
sudo mv ComputeCpp-CE-$(CCPP_VERSION)-x86_64-linux-gnu
/usr/local/computecpp
export COMPUTECPP_TOOLKIT_PATH=/usr/local/computecpp
export LD_LIBRARY_PATH=/usr/local/computecpp/lib
/usr/local/computecpp/bin/computecpp_info
```

The computecpp_info tool should now list your supported devices.

**Notes:**

- If you see "error while loading shared libraries: libOpenCL.so" then you have not installed the
OpenCL drivers needed to run ComputeCpp.
- If the device is listed as "untested" by the tool, Codeplay does not test or support that specific device but it would still be expected to work correctly.

Install Bazel

This version of bazel requires unzip <= 6.0.0 to install.

```
wget https://github.com/bazelbuild/bazel/releases/download/0.16.0/bazel_0.16.0-linux-x86_64.deb
sudo apt install -y ./bazel_0.16.0-linux-x86_64.deb
bazel version
```

Check that the bazel version output from the above command is 0.16.0. More recent versions may work but are not supported.

Configure TensorFlow

```
git clone http://github.com/codeplaysoftware/tensorflow
cd tensorflow
```

The configure step is controlled by environment variables. You can leave them unset to instead get questions prompted on the terminal.

```
export CC_OPT_FLAGS="-march=native"
export PYTHON_BIN_PATH="/usr/bin/python"
export USE_DEFAULT_PYTHON_LIB_PATH=1
export TF_NEED_JEMALLOC=1
export TF_NEED_MKL=0
export TF_NEED_GCP=0
export TF_NEED_HDFS=0
export TF_ENABLE_XLA=0
export TF_NEED_CUDA=0
export TF_NEED_VERBS=0
export TF_NEED_MPI=0
export TF_NEED_GDR=0
export TF_NEED_AWS=0
export TF_NEED_S3=0
export TF_NEED_KAFKA=0
export TF_DOWNLOAD_CLANG=0
export TF_SET_ANDROID_WORKSPACE=0
export TF_NEED_OPENCL_SYCL=1
export TF_NEED_COMPUTECPP=1
```

The following environment variables may differ depending on your device:
export TF_SYCL_PRESET=<preset>

Set a preset to select the best set of options for a specific platforms. Supported presets are:

- AMD_GPU
- INTEL_GPU
- ARM_GPU
- POWER_VR

If this is set you can jump to the Build TensorFlow section. The preset can be overwritten by exporting any of the variables below.

**Note:**

- The ARM_GPU and POWER_VR presets will automatically set the --cpu=arm flag if compiling natively and --cpu=armeabi if cross-compiling. If these presets are not used, either flag needs to be provided to bazel when compiling for ARM.

export TF_SYCL_BITCODE_TARGET=spir64

The possible values for this option are spir32, spir64, spirv32, spirv64 or ptx64 depending on which intermediate language your OpenCL library supports: * ptx64 is for Nvidia GPUs (highly experimental). * On other platforms, check the device properties output by the clinfo command: * In the "Extensions" field, if cl_khr_spir is present, use spirXX, or if cl_khr_il_program is present, use spirvXX. * Substitute "XX" above for the value of the "Address bits" field. Note that issues can arise if the device's "Address bits" value does not match that of the host CPU e.g. a 64-bit CPU and 32-bit GPU.

export TF_SYCL_USE_DOUBLE=1

Set this to 0 if double-precision floating-point operations are not needed, or are not supported by your device. You can verify this by checking for cl_khr_fp64 in the "Extensions" field of the device properties output by the clinfo command.

export TF_SYCL_USE_HALF=0

Half-precision floating-point operations are not supported yet.

export TF_SYCL_USE_LOCAL_MEM=1

This option is not prompted during the configure step. If it is unset, some operations will generate two sets of kernels - using local memory and not using local memory - and the right one will be selected at runtime. It is advised to set this value to "1" if your device supports it. Set it to "0" if the "Local memory type" field of clinfo is not "Local" or if "Local memory size" is smaller than 4KiB.

export TF_SYCL_USE_SERIAL_MEMOP=0
This option is not prompted during the configure step. It is recommended to set it to "0" for better performance. Not all devices support this option. If all the operations using the GPU throw an exception, please set this to "1".

```
export TF_SYCL_OFFLINE_COMPILER=<path/to/compiler>
export TF_SYCL_OFFLINE_COMPILER_ARGS=<args>
```

These options are not prompted during the configure step. If set, ComputeCpp will call this compiler to offline compile the OpenCL kernels. The offline compiler to use depends on the OpenCL implementation. TF_SYCL_OFFLINE_COMPILER_ARGS can be used to provide arguments to the offline compiler. These options require a Professional Edition of ComputeCpp.

Build TensorFlow

Build the pip package with:

```
./configure
bazel build --verbose_failures --jobs=6 --config=sycl --config=opt
//tensorflow/tools/pip_package:build_pip_package
```

Notes:

- Make sure to re-run ./configure if you change any of the environment variables described in this guide.
- It is recommended to keep the exact same environment variables when building again with bazel to avoid re-building from scratch.
- It is recommended to provide --jobs=X to bazel with X strictly smaller than your number of threads to reduce RAM usage.

Bundle and install the wheel

Choose an existing folder to output the TensorFlow wheel, for example:

```
export TF_WHL_DIR=$HOME
```

```
bazel-bin/tensorflow/tools/pip_package/build_pip_package $TF_WHL_DIR
pip install --user $TF_WHL_DIR/tensorflow-1.9.0-cp35-cp35m-
linux_x86_64.whl
```

The .whl file may have a different name depending on your architecture or the Python version provided by PYTHON_BIN_PATH.

Many tests and benchmarks require more pip packages than the minimal set of packages listed in the build pre-requisites. The versions listed below are known to work with this build of TensorFlow:
pip install -U --user numpy==1.14.5 wheel==0.31.1 six==1.11.0 mock==2.0.0
enum34==1.1.6 portpicker==1.2.0
pip install -U --user scipy==1.1.0
pip install -U --user scikit-learn==0.20.2
pip install -U --user --no-deps sklearn

Run the benchmarks

To verify the installation, you can execute some of the standard TensorFlow benchmarks. The example below shows how to run a single inference of AlexNet:

git clone http://github.com/tensorflow/benchmarks
cd benchmarks
git checkout cnn_tf_v1.9_compatible

# Allow the SYCL device as a valid option
sed -i "s/'cpu', 'gpu'/'cpu', 'gpu', 'sycl'/" scripts/tf_cnn_benchmarks/benchmark_cnn.py

python scripts/tf_cnn_benchmarks/tf_cnn_benchmarks.py --num_batches=10 --local_parameter_device=sycl --device=sycl --batch_size=1 --forward_only=true --model=alexnet --data_format=NHWC

Run the tests

Running the tests is a good way to check TensorFlow built correctly. You can run a large set of about 1500 tests with the following command:


Alternatively you can run a subset of the tests above, running only TensorFlow operations with the following:

bazel test --test_timeout 1500 --verbose_failures --jobs=1 --config=sycl --config=opt -- //tensorflow/python/kernel_tests/...

Some of these tests are expected to fail due to using unsupported types. Make sure to use the same
options as during the build step as `bazel test` will re-build the targets with the given options.